游易-程序主题分享合辑

知识管理部 等 2022.06.07 15:37 6237 321 184个资源

汇总从101至今的游易征稿文章合辑。

基于neox地形系统的随机地表生成算法

王惊雷 2018.08.07 17:38 1077 16 1 查看原文

分享一种随机生成NEOX地表贴图的算法思路,设计了一套基于柏林噪声底图+分形迭代调整的混合贴图生成算法，同时采用膨胀，高斯核卷积等数字图像处理的方法来优化贴图边缘表现

1 前言

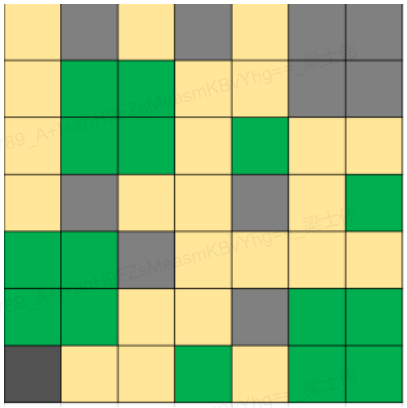
在16年项目初期策划提出一种跟地形系统挂钩的玩法，要求按一定的规则随机生成每一个玩家的地形地图，地图中包括沙地平原和沃土四种地表，同时根据每种地表的面积大小按比例随机生成树木，石头，矿等资源。

该玩法要求每个玩家的地图都不一样，同时地图与玩家的初始资源分布紧密相关，因此玩家地图的地表需要从美术手动绘制转自动生成。

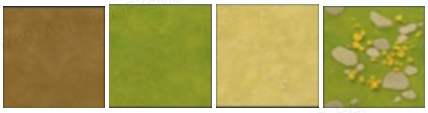
针对这一需求我们做了一些调研，研究了类似《饥荒》《矮人要塞》等游戏的2D随机地图生成算法，发现并没有现成可用的在NEOX地形的框架下满足策划对分布和比例的需求，于是只能自己尝试设计了一套有一定可控性的随机生成流程。

但好景不长，该玩法开发到一半中道崩殂，被迭代掉了，整套方案最终只实现了整体上的分布需求，局部细节没有做完，地表间的颜色过渡和边缘形状的控制没有在游戏里得以实现，算是个半成品。本文旨在把之前游戏里已经实现的部分进行一个总结，生成算法的思路，希望能起到抛砖引玉的作用。

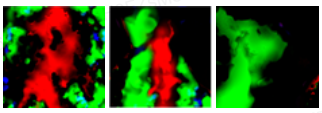
2 需求描述



（1）以NEOX地形系统中的地块(chunk)为基本单位，每个地块上有一种主地貌，如上图所示，绿色代表草地和沃土，灰色代表黄色代表平原。该图由策划填表控制，通过行列告诉第几块chunk应该是什么地貌。



（a）基础地表贴图



（b）场景编辑器中绘制出的混合贴图

（c）纹理混合后产生的最终地表

NEOX的地形系统在不魔改的前提下最多支持4张基础地表贴图，美术同学在场景编辑器里用笔刷绘制地表的过程实质上是生的纹理混合贴图(BlendMap)的过程。地形的shader会去采样地表每个像素点对应的混合贴图颜色值作为基础纹理贴图两两混合的子。比如混合贴图的R通道就是第一张基础纹理和第二张基础纹理的混合因子，G通道是12混合后和第三张混合的因子，B通道是后和第四张混合的因子。

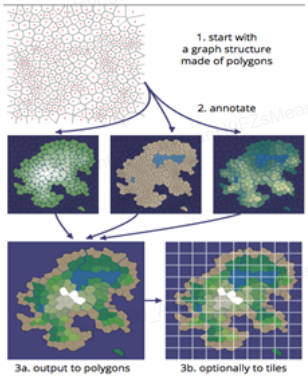
```
branch("TERRAIN_ALPHAMAP_4")
{
     float3 result_1 = lerp(diffuse_map_color0, diffuse_map_color1, float3(blend_factor0));
     float3 result_2 =  lerp(result_1, diffuse_map_color2, float3(blend_factor1));
     out(result,lerp(result_2, diffuse_map_color3, float3(blend_factor2)));
}
```

于是随机地表乃至整个随机地图的问题核心可转化为生成一张与地图尺寸相等的高质量的纹理混合贴图，且该贴图的颜色比例分布符合策划的需求。这里对高质量的定义包括两部分，其一是分布的自然效果符合视觉认知，其二是误差可控，符合比例和空需求。

4 随机地图生成算法调研

(a) MineCraft的3D柏林噪声算法

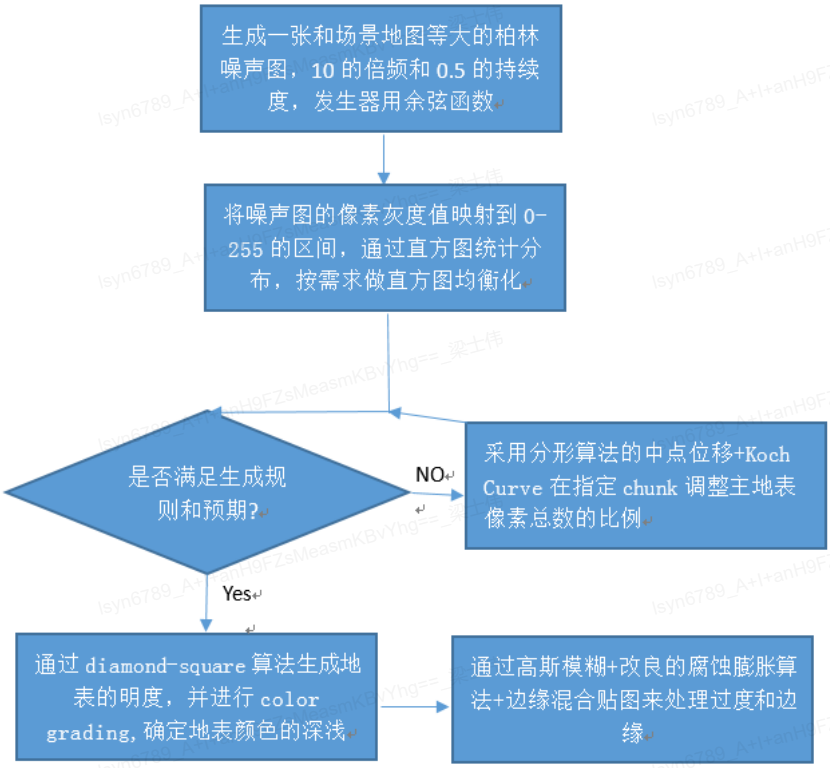
(b)饥荒的随机原点+多次三角函数算法



(c) 基于denaulay三角网格+蓝噪声的岛屿地图生成算法 (d)上古卷轴procedural generation

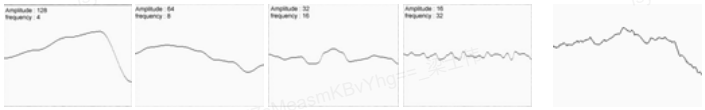
首先是以上古卷轴和地平线为代表的程序过程化生成算法，也是目前3D游戏的主流算法，可惜的是看了下地平线的在GDC上该思路对生成二维的混合贴图的目标帮助不大。

5 算法流程



一句话概括即：用余弦发生器的柏林噪声生成算法制作整体分布底图，用分形算法来调整分布比例和地表的颜色层次，用数学

柏林噪声的生成



(a)柏林噪声的基底　　(b)合成后的噪声函数

柏林噪声的原理简单来说就是对不同振幅和频率的函数进行叠加，得到一个符合需求的噪声发生器。具体的算法细节可以参考libnoise的教程http://libnoise.sourceforge.net/tutorials/tutorial4.html。在本算法中，该噪声发生器的输入是地图上的XY坐标，采用余弦函数进行噪声曲线叠加后的插值计算，取9倍的倍频和0.48的持久值，生成地图大小的柏林噪声图。

噪声图的直方图统计和直方图均衡

策划需要控制不同地表的比例关系，如25%的浅色草地，25%的深色草地，25%的沙地，15%的平地，10%的雪地这样的比例。因此在噪声图上需要将0-255动态划分为符合策划比例的若干个区间，如果分布不均匀，需要对噪声图进行直方图均衡化的操作。

$$s_k = \sum_{j=0}^{k} \frac{n_j}{n} \quad k = 0,1,2,\cdots,L-1$$

上面的公式为直方图均衡化的映射公式，原理就是通过累积分布函数来进行灰度区间的再分配。

经过直方图统计和直方图均衡后，我们根据比例可知0-255的范围内哪些灰度区间对应哪种地表，比如0-64对应草地，64-96对地，96-192对应平地，192-255对应深色草地等等。

分形算法在噪声图的基础上进行调整

纯柏林噪声的分布是无法满足策划对地表控制要求的，因此需要提供针对指定位置指定大小生成符合预期的单块地表的功能。对某个chunk需要对主地表颜色的比例进行调整，分大小两种调整: 小调整就直接在原有的边缘上进行分形，大调整可以直接在指定位置生成完全由分形系统生成的混合块，覆盖到原有的混合贴图上。分形算法在整体分布上起到细化和调节的作用。

二维中点位移过程

尝试了多种分形算法，结果发现还是最简单的中点位移（MidPoint Movment）最为实用，最大的优点是可控性和可伸缩性非理很简单，每次取每段直线的中点，对中点施加一个随机的偏移量，并递归的执行下去，同时偏移量不断减小，最终生成符合需形状。

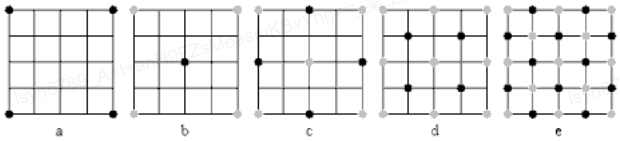
在游戏中我们需要生成的是XY二维的中点位移，且需要对偏移范围的衰减函数进行多次调整取得一个较好的经验值，才能生高的随机形状。如果偏移范围衰减太快和太慢，都容易出现突刺问题或者变化较小比较呆板的问题，如下图所示。所以中点位移心还是衰减函数的设计和衰减函数的参数调整。

（a）衰减太快，有边缘突刺产生　（b）衰减太慢，看起来变化太少很呆板

在设计衰减函数方面我们暂时也没太多经验，最后试了一系列函数后，采用Sigmoid函数来计算每次的衰减值的base,再通过一个经验因子，作为第N次衰减的最终因子值。

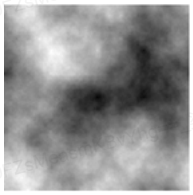
（2）diamond-square算法

用DS算法的目的主要是对草地的深浅做出调整，体现出地表颜色的层次感。同时玩法设计中沃土是必须生成在草地上的，所深浅颜色对两种地表进行区分。



ds算法的明度差值步骤

分形算法产生的图案具有局部的自相似性，因此经常用来生成如火焰，云，海水等特效的纹理采样图。这里用于生成地表贴图值。DS算法的原理也相对简单，一共分为两个步骤:diamond step: 先用四个顶点+随机扰动生成中心点的值。square step: 计算构成的菱形的中点，同样是四个顶点均值+扰动值，再递归执行第一步即可，如上图所示。通过DS可以生成如下所示的灰度图，color grading后，每个像素点映射回原混合贴图的明度值，即可产生颜色的层次效果。



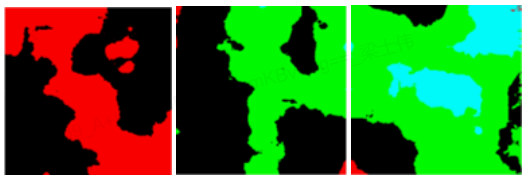
算法生成的明度分布图

我们采用分形的中点位移算法是为了调整主地表边缘的形状和控制分布比例，采用diamond-square算法生成明度图是为了调内地表的颜色亮度和层次。

生成的地块效果图

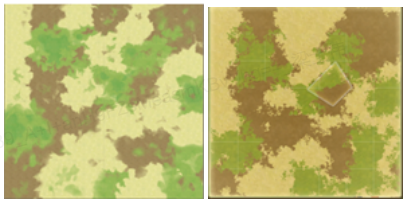
最终效果整体分布还可以接受，但是在不同地表边缘过渡和颜色细节上很遗憾还是没来得及做出预期的效果。

（1）混合贴图生成

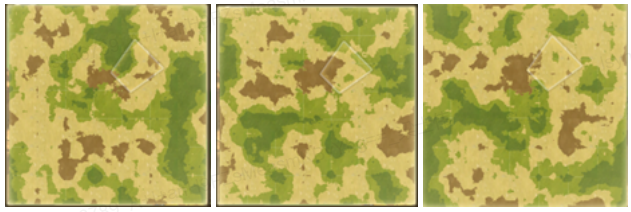


算法生成的混合贴图形状和边缘

（2）地表分布效果



美术原画的分布图　　程序生成的分布图



不同随机种子生成地表分布图（7x7chunk）

经过数据统计生成出来的各种地表比例的误差基本可以控制在10%以内，满足策划的需求。

（3）局部细节图对比

美术原画　　程序生成地表

因为没有做好边缘形状和颜色的过渡，所以可以看出生成的边缘还是硬边，交界处很突兀。

7 总结和展望

在经过尝试后，个人认为程序化生成混合贴图的方向是可行的，只是目前给出的方案存在以下三个问题：

（1）无监督的随机生成不可行，还是需要生成指导和生成质量评估的量化。目前这套方案欠缺对生成贴图质量的评 约束例，但对生成出来的视觉效果评价没有建立评估公式。建立这样的评估和约束难度也比较大，这个也是整个随机过 要 目标，可能需要进一步的算法调研。有了这一度量标准，我们才能判断哪些是好的随机结果，哪些是不好的随机结 还 去筛选。

（2）没有动态添加生成条件的功能，美术和策划难以介入随机过程。实际上这个生成器还是应该做成一个可交互的 而自动化的一个算法。它的定位应该是一个提升场景制作生产效率的工具，完全依赖随机还是过于理想化了，随机发生器的因子，减因子等参数都应该暴露出来在编辑器中可配置，这样生成出来的地表贴图才可控。

（3）局部细节（包括边缘形状和颜色的过渡，多地表的分布混合等）还需要进一步花时间改进算法。

最近看了siggraph17一篇基于用户指导的图像着色算法(https://richzhang.github.io/ideepcolor/),想起来16年做的这个随机地表其实是需要user-guide的部分去提升混合贴图生成的质量的。另外特别是如果能加入一些场景重建和图像学习的东西进去，通过地貌的数码照片来影响生成器的因子，可能会更有意思一些。希望以后有其他的机会可以继续完善局部细节的处理和尝试新的

[5]http://www-cs-students.stanford.edu/~amitp/game-programming/polygon-map-generation/

[6]http://libnoise.sourceforge.net/tutorials/tutorial4.html

[7]http://www.gulu-dev.com/post/2014-11-16-open-world

[8]PSMAGE: Balanced Map Generation for StarCraft,Alberto Uriarte,Santiago Ontañón,2013 IEEE Conference on Computational Inteligence in Games (CIG)

全部评论 1

请输入评论内容

还可以输入

(可添加1个视频+5张图片) 匿名

最热 最新

匿名

1楼

2018-11-14 21:04

加载完毕,没有更多了